Designing and comparing access control systems

Hannes Boberg

Supervisors: Bernhard Thiele (IDA), Paul Engström (Pdb)
Examiner: Martin Sjölund
Upphovsrätt


Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances. The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page [http://www.ep.liu.se/](http://www.ep.liu.se/).

© Hannes Boberg
Abstract

Access control systems are an important concept in the area of computer security. In this master thesis different solutions are analyzed. The focus is on a tool called DW Access. DW Access is developed by Pdb Datasync AB. A comparison was done that showed that DW Access is lacking some important functionality. After the comparison a base model for an access control system was designed. The new design includes concepts like relationships, replacements and time limited access. It also works for generic subjects and objects in the system. This design was later partly implemented in DW Access.

The conclusions from this thesis work is that DW Access is a unique tool and there is a market for the application or similar applications. The new functionality was one step forward and the evaluation showed that the potential users liked the new concepts. But it is a very open area because of very unique requirements on the market.
I would like to say thank you to my supervisors and my examiner for all help and input you have given me in this thesis work. I would also like to say thank you to Pdb for giving me the chance and resources to do this very exciting thesis work.

I would also like to say a big thank you to all my friends and my family. You have been a huge support during my whole education. Without you I would not be where I am today and I am forever grateful for that!
Contents

Abstract iii
Acknowledgments iv
Contents v
List of Figures vii
List of Tables viii
Acronyms 1

1 Introduction 3
  1.1 Motivation 3
  1.2 Aim 3
  1.3 Research questions 3
  1.4 Delimitations 4

2 Theory 5
  2.1 Access Control 5
    2.1.1 Policies 6
    2.1.2 Data structures 6
    2.1.3 Access operations 7
  2.2 Business Intelligence 8
    2.2.1 Multidimensional cubes 8
    2.2.2 Data warehousing 8
  2.3 Structured Query Language 8
  2.4 Microsoft SQL Server 9
  2.5 Microsoft Power BI 9
    2.5.1 Gateways 9
    2.5.2 Row-level security 9
  2.6 Microsoft Azure 9
  2.7 Active Directory 9
  2.8 Lightweight Directory Access Protocol 9

3 Method 11
  3.1 Pre-study 12
  3.2 Comparison 13
  3.3 Design 13
  3.4 Implementation 15
    3.4.1 Row-level Security in Azure SQL 15
    3.4.2 Setting up the filter database demo environment 15
    3.4.3 Stored procedures and functions in demo environment 17
List of Figures

2.1 A generic model of an access control matrix ........................................... 7
2.2 A generic model of an access control list for object o ................................ 7
2.3 A generic model of a capability for subject s ........................................... 7

4.1 The UI for setting filters for groups in DW Access .................................... 27
4.2 The different layers of DW Access and how they communicate ................. 28
4.3 The different data layers of DW Access .................................................. 29
4.4 The flow when a user is requesting filtered data in DW Access ................. 29

5.1 Entity-relationship diagram for the proposed model ................................. 34
5.2 The new view for filtering ................................................................. 34
5.3 The right click menu on a filter item ..................................................... 35
5.4 The view for setting up or edit a replacement between users .................... 35
5.5 Flowchart for adding a user or a group ................................................. 36
5.6 Flowchart for adding a template to a user or a group ............................... 36
5.7 Flowchart for the function to check a filter .......................................... 37
5.8 The UI for filtering in the merged version. Also the templates can be seen. 38
5.9 The UI for creating a new replacement ............................................... 38
List of Tables

2.1 A comparison of four different access operations....................... 8
5.1 A comparison between the different software presented in the report..... 33
5.2 Requirement specification.................................................. 33
5.3 Result of the survey in the evaluation part............................. 37
Acronyms

ACL  access control list.
ACM  access matrix.
AD   Active Directory.
API  application programming interface.
BI   business intelligence.
DAC  discretionary access control.
DBMS database management system.
DLL  dynamic-link library.
DWA  DW Access.
JDM  Jet Data Manager.
MAC  mandatory access control.
MSSQL Microsoft SQL Server.
NIST National Institute of Standards and Technology.
OLAP online analytical processing.
Pdb  Pdb Datasystem AB.
RBAC role-based access control.
ReBAC relationship-based access control.
Acronyms

**RLS**  Row-Level Security.

**RuBAC**  rule-based access control.

**SQL**  Structured Query Language.

**SSAS**  Microsoft Analysis Service.

**TX DWA**  Timextender Data Warehouse Automation.

**UI**  user interface.
1 Introduction

1.1 Motivation

In large companies the amount of sensitive data is often huge and the data availability is very important. But it is also as important that no non-authorized user can access the data. This can be an internal requirement set by the policy at the company but it can also be a requirement by legislation. Also data integrity is a very high priority. To make sure the policy is fulfilled an access control system is often used. An access control system is a layer between the user and the data that makes sure that the data access is allowed within some sort of policy. In the last years data storage has changed from mainly on-premise solutions to more cloud based solutions, according to Pdb Datasystem AB (Pdb). Hybrid solutions are very common where data is both in the cloud and on-premise. This has lead to an increased demand for more platform independent access control solutions. The question is how to design those systems but sustain the high security requirements that is of huge priority. At the same time it shall not impact the usability more than necessary. This master thesis will take a look at this and make an analysis from the perspective of the end user.

1.2 Aim

The aim of this master thesis is to compare different on-market solutions and how well they can be suited for companies in need of access control. After this a design suggestion based on a reference system will be presented and included in the comparison. The comparison will be based on requirements and reviews by the end users and theory from the research in the area.

1.3 Research questions

1. How to design an access control system for hybrid solutions where many different permissions and underlying platforms must be considered.

2. How is the situation on the market today?

3. How to make sure that the layer of access control does not affect the usability more than necessary.
4. What is the forecast for the future in the area of access control?

1.4 Delimitations

- All implementations will be done in C# and Microsoft SQL Server (MSSQL).
- The thesis work will focus on a particular hybrid solution.
- Only parts needed for evaluation will be implemented.
- The implementation will be based on MSSQL and Azure SQL.
- The implementation will be based on the code base in the access control system DW Access (DWA) developed by Pdb.
- The focus will be on systems for business intelligence (BI).
In this chapter the theory about the topics in this thesis will be presented. It will give the reader the base knowledge and terminology that is required to understand the rest of the thesis. The things that will be introduced are the concepts of access control and BI. Also the Structured Query Language (SQL) and some systems that will be examined in this thesis will be introduced briefly.

2.1 Access Control

The meaning of access control in computer systems is the process of deciding who is eligible to access certain information [1]. The difference between access control and just denying access to the whole system is that access control systems are used to limit the access for users that are legitimate in parts of the system. Even programs that are executed by the users need to be granted access to the information they are trying to access by the access control system [2].

The National Institute of Standards and Technology (NIST) has found out that the confidence of customer information is a critical aspect for many organizations. They even got told that it was even more important than providing a good service at a reasonable price by the vice president of a major bank [3]. It can even be a requirement by legislation. One example of this is sensitive personal data that is stored in the healthcare industry [4].

In computer systems there are passive entities called objects that are holding or receiving information. It also has entities called subjects that can be users or programs that are trying to access the objects. The access control is used to grant or deny the access. This is also called authorization [1], [2].

To be able to do this authorization the system needs to know the identity of the entity that is asking for the access. The process to verify the identities of entities is called authentication. Authentication is very important in the sense of access control because if the access control does not know who is asking it cannot decide whether it is an authorized request or not [1], [2].
2.1. Policies

An access control policy is a high level description of how the access control shall determine if an access is eligible or illegible [2]. Even though most organizations see their needs as unique some basic policies are needed [3]. In this part some of these policies that will be discussed in this thesis will be introduced.

Mandatory Access Control

Mandatory access control (MAC) is a policy that is based on a central administrated classification on objects and subjects. Every entity in the system has a security level. The security level is based on how sensitive the information is. The relationship between the security levels decide if the access is granted or not. Often the requirements are different depending on the type of access [2].

A common access policy with MAC is based on a security theorem that was developed by D.E. Bell and L.J. LaPadula. This theorem states that read access is only granted to objects with lower or equal security level and write access is only granted to objects with higher or equal security level. This is to make sure no information is leaking downwards in the hierarchy. This is the principles of read down and write up [2], [5].

Discretionary Access Control

Discretionary access control (DAC) is a policy where the identity of the entities is used to decide if the access is eligible or not. The policy can either be set in a centralized manner where the system owner, or a subject decides who has access to objects or it can be decentralized where the owner of the object can decide who is eligible to access the information [2].

Role-based Access Control

With role-based access control (RBAC) the functional role (for example a role in an organization) is used to determine what permissions the subject has. RBAC is based on a model where the roles within an organization is relatively persistent and therefore RBAC will lower the overhead and cost of assigning permission when for example a new employee is introduced in the system [6].

Rule-based Access Control

Rule-based access control (RuBAC) is a generic policy for access control where the rules decided by the system owner determine the permissions for the subjects. As there is no commonly understood definition of rules this is a very broad category of access control systems [7].

Relationship-based Access Control

With relationship-based access control (ReBAC) the relationship between the subjects determine the permissions to objects. In many organizations roles are not sufficient and attributes to the roles have been added. With ReBAC the policy is based on this relationship instead [8].

2.1.2 Data structures

This section will introduce the data structures that are often used for access control system implementations. At the beginning some important data types will be introduced.
2.1. Access Control

Access Matrix

An **access matrix (ACM)** is a data structure where there are subjects in the rows and objects in the columns. Every access right is then found in the matrix by finding the intersection. One example is when some subjects $s_1 \ldots s_n$ are defined together with some objects $o_1 \ldots o_m$ and for each couple $(s_x, o_y)$ a permission $p_{x,y}$ is defined. This would give the ACM seen in Figure 2.1 [9]. The biggest issue with ACMs is that in a normal system it will be enormous. Also, many cells will be empty so the solution is also not very memory efficient. It is therefore not widely used in practice in this form [2].

\[
\begin{array}{cccc}
  o_1 & \cdots & o_m \\
  s_1 & p_{1,1} & \cdots & p_{1,m} \\
  \vdots & \vdots & \ddots & \vdots \\
  s_n & p_{n,1} & \cdots & p_{n,m}
\end{array}
\]

Figure 2.1: A generic model of an access control matrix.

Access Control List

An **access control list (ACL)** is a data structure where every object in the system holds a list with users and their permissions. A generic example of an ACL is shown in Figure 2.2 where we have an object $o$ and subjects $s_1 \ldots s_n$ and permissions $s_x$ has on $o$ as $p_x$ [2].

\[
o \rightarrow [p_1, \cdots, p_n]
\]

Figure 2.2: A generic model of an access control list for object $o$.

Capabilities

Capabilities is a data type where every subject has a list with the permissions it holds to all objects. A generic example of a capability is shown in Figure 2.3 where we have a subject $s$ and objects $o_1 \ldots o_n$ and the permissions $s_x$ has on $o_x$ as $p_x$ [2].

\[
s \rightarrow [p_1, \cdots, p_n]
\]

Figure 2.3: A generic model of a capability for subject $s$.

2.1.3 Access operations

In a computer system we can either observe (look at the object) or alter objects (change the object) [10]. In this section four access operations will be introduced. A comparison of these access operations is seen in Table 2.1.

**Read**

Read can be described as the access right that observes but not alters the data [10].

**Write**

Write can be seen as the access right that alters and observes the data [10]. Usually write access implies read access [11].
2.2. Business Intelligence

Append

Append can be seen as the access right that alters the data but does not observe it [10]. Append means a write access that does not imply read access [11].

Execute

Execute is a data access that neither alters nor observes the data [10]. Although many implementations require read rights to execute it is not always the case [11].

Table 2.1: A comparison of four different access operations.

<table>
<thead>
<tr>
<th>Access operation</th>
<th>Read</th>
<th>Write</th>
<th>Append</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alter</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Observe</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

2.2 Business Intelligence

BI are different applications and techniques to make decisions for business users with the help of data. The systems can for example gather and analyze the data and provide it in a form that helps the user making the right decision. It can be described as a value chain where data is transformed into information for the users of the system [12].

2.2.1 Multidimensional cubes

A cube is data stored in multidimensional arrays where the cells are grouped by dimensions. A dimension is based on the user’s perception of some attribute of the data. A dimension can be hierarchical and the data can be viewed in different levels of details based on the hierarchy. One example can be that if time is a dimension it can be split into years, months or days. Multidimensional means that the cube can be stored based on several different dimensions [13].

2.2.2 Data warehousing

Data warehousing is a number of technologies to store data for decision support. Data warehouses store data in a form that is best for the analysis that is done on the data. One example can be that every individual transaction is of no interest and instead it stores some sort of summary of all transactions, something often referred to as online analytical processing (OLAP). Data warehouses often extract data from multiple operational databases. The warehouse is then cleaning and transforming the data to the desired format and eventually refreshing what is shown to the end user [14].

2.3 Structured Query Language

SQL is a language for querying relational databases. It was developed by IBM after a paper by E.F. Codd that stated that users of large data banks must be protected to have to learn how the data is stored internally in the system [15]. In 1974 IBM developed the query language Sequel that later became SQL. SQL is a language based on the English language for making queries in relational databases and has become the standard language for relational databases. It can be used both to fetch data and to manipulate data [16],[17].
2.4 Microsoft SQL Server

Microsoft SQL Server is a database management system (DBMS) that is using the SQL language. The DBMS is developed by Microsoft and is the foundation for Microsoft’s data platform. The latest version is MSSQL 2016 [18].

In MSSQL, Row-Level Security (RLS) was introduced in the 2016 version. With RLS, the administrator can allow or disallow users to reach certain data. One example is to let users see the data according to some policy. This can for example be data from the same department as a specific employee is working at. It can be seen as the server automatic inserts a WHERE clause in the query before returning the data to the user [19].

2.5 Microsoft Power BI

Microsoft Power BI is a suite of cloud based tools for BI developed by Microsoft. It can be used to unify data in the organizations when using different platforms. It can create platforms and show data to the users in the way that the users think about the data by creating reports and dashboards from the data [20].

2.5.1 Gateways

Gateways are tools to let users connect Power BI to on-premise data in the organization and refresh the datasets from this on-premise data. The gateway also provides authorization and makes sure that the user is allowed to access the data [21].

There is a personal and an enterprise version. In the enterprise version it is possible for the administrator set up central gateways. It lets the administrator monitor how the gateway is used and for example see the most accessed data sources by the gateway [22].

2.5.2 Row-level security

If Power BI can use RLS or not is dependent on the underlying data. Today it is possible with on-premise data via the enterprise gateway or for cloud models. It lets the administrator choose roles and put restrictions based on some columns in the data [22].

2.6 Microsoft Azure

Microsoft Azure is a collection of cloud based services developed by Microsoft. For example it is used for data storage, analysis and computing. Azure provides all this as services and it is possible for the user to combine this with on-premise solutions [23].

2.7 Active Directory

Active Directory (AD) is a network directory developed by Microsoft. It is keeping track on shared network resources by storing them as objects. An object is typically a user, a printer or some other physical object that is a resource on the network [24]. Resources are grouped together by the group feature. This can reflect the role the person has by some meaning and by this feature it is easier to administrate as every member of a group can be changed simultaneously [25].

2.8 Lightweight Directory Access Protocol

Lightweight Directory Access Protocol (LDAP) is a protocol to communicate with existing directories. It is running over the TCP/IP stack and provides an application programming interface (API) for simplifying writing network applications that require access to a directory.

It features a limited amount of functions compared to the functionality of the directories. These features are intentional, in order to provide a simplified API [26].
3 Method

This chapter is written to introduce the reader about how the thesis work was actually performed. The first thing that was done was to formulate the main questions for the thesis. After an initial meeting with the supervisors and the examiner a planning report was written. The next step was to study literature about the subjects that are introduced in this thesis. The main result of this part of the work is found in chapter 2. The theory part also gives insight about different approaches and techniques that was considered later in this thesis work.

After the theory was done it was time to look at the reference system. The reference system was DWA and is introduced in chapter 4. After an initial meeting with the supervisor a demo environment was configured. This made it possible to have a look at the system. The meeting introduced some more guidelines:

- The main platforms for the model should be:
  - Microsoft Azure
  - MSSQL
  - Microsoft Power BI

- The work is mainly on access control in data warehouses. Also some questions was raised in this meeting:
  - How are the access control systems working today for the end user in large and medium large companies?
  - What can be done directly in the systems? When is there a need for a third party system?
  - How can the automation be increased in the access control?
  - What is the purpose of DWA?
  - How well is it performing compared with its competitors?
3.1 Pre-study

The pre-study started with a market analysis to identify other software with similar purpose as DWA. The market analysis were done by searching for similar software and deciding if they was a competitor or not. The software systems that was found in this phase was:

- Jet Data Manager (JDM)
- Timextender Data Warehouse Automation (TX DWA)

All the different software systems are introduced in chapter 4.

Some own ideas about what to investigate was discussed and introduced:

- Time limited permissions.
- Transferred permissions.
- Relationship between users/groups (Some sort of ReBAC).
- Integration with HR solutions.

The next step was to get the end user perspective of this ideas. A survey with questions was sent to people with insight about DWA. Internal documentation with a status report from one of the customers was also used. From the status report some new ideas were introduced. One was time limited permissions. Another was the concept of replacements. One example of the latter is when an employee is leaving for parental leave and another employee is replacing this employee in the organization. This was added to the list of things to investigate in this thesis work. A demo environment from Pdb was set up at this stage. This environment was used to gain insight about the data structures behind the access control more deeply. This environment was used together with the source code to gain insight about the software. More about the software can be read in section 4.1.1.

The main inspiration came from investigating the system and asking for input from people with insight. Also comparing to other systems was important. The following ideas for the access control policy setting was found interesting:

- Time limited permissions.
- Transfer permissions to other users.
- Users replacing other users for example when they are on parental leave.
- Explicitly deny permission that is denying a subject to an object even if another permission says the access is allowed.
- Combination of individual and group permissions in the same model and software.
- Expandable to more platforms via modules.
- An API for developing modules.
- Relations between users.

The result of the survey and the status report earlier mentioned was the main tool for the rest of the design work. It also provided guidelines when deciding what to look at in the comparison between the different software systems.

The pre-study also confirmed that the requirements on an access control system are very unique. Functionality that was requested by one user was of no interest at all for another. As the mission was to design a competing system the decisions about further limitations was made after the comparison.
3.2 Comparison

Based on the pre-study part some functionality that was worth comparing was decided. The following functions were decided to be used in the comparison.

- **Row-Level Security**
- Views.
- Differentiated read and write permissions.
- A deny choice.
- Time limited filters.
- User replacements.
- Set permissions on users/groups/roles in the same software.

More about those functions can be read about in section 4.2.

After that the available documentation for the different software was investigated. Manuals for both TX DWA and JDM was found by the software publishers. For DWA some documentation that Pdb had for internal usage was used. This documentation was used to distinguish the functionality in the list. In DWA the functions could be tested instead. Also input from some developers who were working with DWA were used in this step. The result of the comparison can be seen in Table 5.1 on page 33.

3.3 Design

The design part was started with investigating how DWA is designed. A brainstorming session based on the results from the pre-study and the comparison was the next step. The design should include both functionality that the competing software included. It should also include some new functionality that the end users of DWA had asked for. The most important part was to merge the two versions together. That meant making it possible to both set permissions based on single users and groups of users. The design should also be able to do access control on more data storage systems like MSSQL and Microsoft Azure for example.

The first step was to create a requirement specification that can be found in Table 5.2 on page 33. After this the database design for the two different versions of DWA was studied. Next, some ideas about how to implement the functionality was proposed.

- Time limited filters can be implemented by including columns for start time and end time. If those fields are NULL-valued no start time or end time is set.
- In MSSQL 2016 the feature for RLS can be used. It could either be done by connecting the security policy to the filter database or it could be done by having a separate filter table. Another approach would be to dynamically create a new policy when a new filter should be implemented.
- Views can be used in older versions of MSSQL. This can be done either by adding tables for filtering or by connecting the views to a filter database. The view is used to only show the data that shall be seen to the user.
- In Power BI the access control is handled on the underlying data layer.
- Filter items for users and groups can be stored in the same way. The procedures to check filters is then adapted to this. This would make it possible to merge the code of the different versions of DWA with just minor changes.
• Every filter item can hold a binary value for read or read/write (no write without read is included).

• Another binary value is added to the filter items to tell if it is allow or deny. If the value shall be not set no filter item is created in the database.

• Another approach can be to use differentiated filter types stored in the database. This can be used both for the allow/deny and the read/write filtering. This would also make the design more expandable.

After this the design phase for the new database was started. The result of this work can be seen as an ER-diagram in Figure 5.1. The new features in the model are:

• All subjects have a generic id called filter id. This let us store every filter item in the same table.

• A table for replacements is introduced. The replacement table holds two users, a start date and an end date. A NULL-valued date for a replacement date means it is not set. This lets the administrator set a user as a replacement for a user. The other user will then inherit the permissions of the other user.

• Support for relationships between users is included in the model. One table for storing relationships and one to store different types of relationships is introduced.

• Different filter types for making the distinction for read and read/write permissions. They also distinguish between allow and deny. A more generic solution is provided though so that the model supports an arbitrary number of different filter types defined in the implementation of the system.

Next step was to make sure that the model is working on hybrid solutions. The focus should be:

• MSSQL databases

• Multidimensional Microsoft Analysis Service (SSAS) cubes.

• Azure SQL.

The approach for this was different for these versions. The SSAS-cube already solves this by a stored procedure in SQL. So here the solution was to implement necessary changes for this procedure based on the new model. For MSSQL two approaches were used based on the running version of MSSQL:

• In the 2016 version the functionality for RLS is used. It can either be used with a filter database, a filter table, or dynamic policies.

• In earlier versions unique views are created for each filtered table. Another table in the database holds the filter settings. The subjects then only have access to the views and not the original tables. Also the filter database can be used.

• Azure SQL uses the same solution as MSSQL 2016.

The resulting database model for the filtering that was developed in this design phase can be found in section 5.3.
3.4 Implementation

The first step in the implementation process was to set up a demo environment for the filter database to be able to test some ideas. This demo environment was a copy of the proposed model in Figure 5.1 implemented in MSSQL. The database was filled with some test data that was used to try the ideas before implementing it on the slightly more complex database for DWA. A demo database with data to test RLS in MSSQL via Azure SQL was also set up.

The first thing to implement on the demo database was to combine the filtering of users and groups and implement the replacement functionality. This was done using stored procedures and user-defined functions.

3.4.1 Row-level Security in Azure SQL

In Azure SQL a demo database was created to test RLS. This was done by creating three tables:

- Test data
- Levels
- Filter table with the users and their allowed levels.

Every row in the test data had an associated level. Some random test data and levels is added. The procedure to implement the RLS was:

1. Create a function that returns 1 if the active user has the requested level in the filter table.
2. Add the function as a security policy to the table with test data.
3. Add a user and give it access to the test data.
4. Try to read from test data. Nothing is seen.
5. Give the user access to a level in the filter table.
6. Try to read data from test data again. Now the data with the right levels are shown.

3.4.2 Setting up the filter database demo environment

In this subsection the implementation of the model provided in Figure 5.1 is provided. The implementation was done in MSSQL 2016. Relations and relationships table are not included as motivated in section 6.2.4. The database was implemented using the following query to create the tables and the foreign key constraints:

```sql
CREATE TABLE Levels
(
    id int IDENTITY(1,1) PRIMARY KEY,
    parentLevelId int
);

CREATE TABLE FilterTypes
(
    id int IDENTITY(1,1) PRIMARY KEY
);

CREATE TABLE Replacements
(

```
3.4. Implementation

```sql
CREATE TABLE FilterItems
(
id int IDENTITY(1,1) PRIMARY KEY,
filterId int NOT NULL,
levelId int NOT NULL,
startDate datetime,
endDate datetime,
filterTypeId int NOT NULL
);

CREATE TABLE Users
(
id int IDENTITY(1,1) PRIMARY KEY,
filterId int NOT NULL
);

CREATE TABLE Filters
(
id int IDENTITY(1,1) PRIMARY KEY
);

CREATE TABLE Templates
(
id int IDENTITY(1,1) PRIMARY KEY,
filterId int NOT NULL
);

CREATE TABLE Memberships
(
id int IDENTITY(1,1) PRIMARY KEY,
userId int NOT NULL,
groupId int NOT NULL,
startDate datetime,
endDate datetime
);

CREATE TABLE Groups
(
id int IDENTITY(1,1) PRIMARY KEY,
filterId int NOT NULL
);

ALTER TABLE Levels
ADD FOREIGN KEY (parentLevelId)
REFERENCES Levels(id);
```
3.4. Implementation

ALTER TABLE Replacements
ADD FOREIGN KEY (substituteId)
REFERENCES Users(id);

ALTER TABLE Replacements
ADD FOREIGN KEY (substitutedId)
REFERENCES Users(id);

ALTER TABLE FilterItems
ADD FOREIGN KEY (filterId)
REFERENCES Filters(id);

ALTER TABLE FilterItems
ADD FOREIGN KEY (levelId)
REFERENCES Levels(id);

ALTER TABLE FilterItems
ADD FOREIGN KEY (filterTypeId)
REFERENCES FilterTypes(id);

ALTER TABLE Users
ADD FOREIGN KEY (filterId)
REFERENCES Filters(id);

ALTER TABLE Templates
ADD FOREIGN KEY (filterId)
REFERENCES Filters(id);

ALTER TABLE Groups
ADD FOREIGN KEY (filterId)
REFERENCES Filters(id);

ALTER TABLE Memberships
ADD FOREIGN KEY (userId)
REFERENCES Users(id);

ALTER TABLE Memberships
ADD FOREIGN KEY (groupId)
REFERENCES Groups(id);

Listing 3.1: Create the tables in the demo database

3.4.3 Stored procedures and functions in demo environment

The following stored procedures and functions were implemented to read and write to the demo database. This section also describes how the functionality was implemented. If not explicitly set in the function or procedure the variables shall implicitly be seen as arguments.

Add levels

Here the parent level must be considered in the implementation. If the level has a parent level it is included as an argument. The parentLevelId will be set to the id of the parent level. If there is no parent level this field will be set to null. After this an insert statement is executed.
3.4. Implementation

to add the new level to the database. From now levels will be referred to as the objects of the system.

```
INSERT INTO levels (parentLevelId)
VALUES (@parentLevelId);
```

Listing 3.2: Add a level

**Add users, groups and templates**

The implementation for users, groups and templates are very similar. In the rest of this section those will be referred to as subjects if not explicitly mentioned. The procedure to add a new subject to the database is the following:

1. Create a new filter item
2. Insert the new subject into the database with the id for the filter item created in the previous step as filterId.

A flowchart for adding a user or a group can be seen in Figure 5.5.

```
INSERT INTO Filters
OUTPUT inserted.id INTO @newFilter
DEFAULT VALUES;

SELECT @filter = id
FROM @newFilter;

INSERT INTO Users (filterId)
VALUES (@filter);
```

Listing 3.3: Add a user

```
INSERT INTO Filters
OUTPUT inserted.id INTO @newFilter
DEFAULT VALUES;

SELECT @filter = id
FROM @newFilter;

INSERT INTO Templates (filterId)
VALUES (@filter);
```

Listing 3.4: Add a group

```
INSERT INTO Filters
OUTPUT inserted.id INTO @newFilter
DEFAULT VALUES;

SELECT @filter = id
FROM @newFilter;

INSERT INTO Users (filterId)
VALUES (@filter);
```

Listing 3.5: Add a template
3.4. Implementation

Add filter types
Filter types are added by an insert statement to add the new entity into the database.

```sql
INSERT INTO FilterTypes DEFAULT VALUES;
```

Listing 3.6: Add a filter type

Add filter items
Filter items are implemented in the following way:

1. Find the filterId for the subject.
2. If no start date is provided as an argument set it to the current time.
3. If no end date is provided as an argument set it to null.
4. Insert the new filter item into the database with the fields set according to the previous steps and arguments to the procedure.

```sql
SELECT @filterId = filterId
FROM Users
WHERE id = @userId;

IF @startDate IS NULL
    SET @startDate = GETDATE()

INSERT INTO Users (levelId, filterId, startDate, endDate)
VALUES (@levelId, @filterId, @startDate, @endDate);
```

Listing 3.7: Add a filter item

Delete filter items
To delete a filter item the end date of the entity is set to the current time.

```sql
UPDATE FilterItems
SET endDate = GETDATE()
WHERE id = @id;
```

Listing 3.8: Delete a filter item

Add replacements
Replacements are added in the following way:

1. If no start date is provided it is set to the current time.
2. If no end date is provided it is set to null.
3. Insert the replacement into the database with the fields set according to the previous steps and arguments to the procedure.
3.4. Implementation

IF @startDate IS NULL
    SET @startDate=GETDATE()

INSERT INTO Replacements(substituteId,substitutedId,startDate,endDate)
VALUES(@substituteId,@substitutedId,@startDate,@endDate);

Listing 3.9: Add a replacement

Delete replacements
Replacements are deleted by setting the end date of the entity to the current time.

UPDATE Replacements
SET endDate = GETDATE()
WHERE id=@id;

Listing 3.10: Delete a replacement

Add memberships
Memberships are added in the following way:

1. If no start date is provided it is set to the current time.
2. If no end date is provided it is set to null.
3. Insert the membership into the database with the fields set according to the previous steps and arguments to the procedure.

IF @startDate IS NULL
    SET @startDate=GETDATE()

INSERT INTO Memberships(userId,groupId,startDate,endDate)
VALUES(@userId,@groupId,@startDate,@endDate);

Listing 3.11: Add a membership

Delete memberships
To delete a membership the end date of the entity is set to the current time.

UPDATE Memberships
SET endDate = GETDATE()
WHERE id=@id;

Listing 3.12: Delete a membership

Assign a template to a user or a group
To assign a template to a user or a group the following stored procedure is used.

SELECT @templateFilterId = filterId
FROM Templates
WHERE id = @templateId
3.4. Implementation

IF @groupId IS NOT NULL
SELECT @subjectFilterId = filterId
FROM Groups
WHERE id = @groupId

IF @userId IS NOT NULL
SELECT @subjectFilterId = filterId
FROM Users
WHERE id = @userId

INSERT INTO FilterItems
(filterId, levelId, startDate, endDate, filterType)  
SELECT (@subjectFilterId, levelId, startDate, endDate, filterType)
FROM FilterItems
WHERE filterId = @templateFilterId;

Listing 3.13: Add a template

Check filter

This is a more complex function than the rest and a flowchart is provided in Figure 5.7. It is implemented in the following way:

1. If the date argument is null it is set to the current date.
2. The filterId for the user and groups of which the user is a member of is added to a table.
3. Check if the object is denied for any of the filterId in the table from step 2.
4. Check if any parent to the object is denied.
5. If it is denied the function stops and returns denied.
6. If it is not denied, get all filterIds for users the user is replacing.
7. Check if the object is allowed for any of the subjects.
8. Check if any parent to the object is allowed.
9. If it is allowed, return allowed. If it is not allowed denied shall be returned.

IF @date IS NULL
SET @startDate=GETDATE()

INSERT INTO @filterIds
SELECT filterId
FROM Users
WHERE id=@userId

INSERT INTO @filterIds
SELECT filterId
FROM Groups
WHERE id IN (SELECT groupId FROM Memberships WHERE userId = @userId
AND (startDate <= @date OR startDate IS NULL)
AND (endDate > @date OR endDate IS NULL));
3.4. Implementation

SET @tempLevel = @level

INSERT INTO @deniedTable
SELECT id
FROM FilterItems
WHERE filterId IN (SELECT id FROM @filterIds)
AND filterType = @deniedType;

WHILE @tempLevel IS NOT NULL
BEGIN
    IF EXISTS (SELECT * FROM @deniedTable WHERE levelId = @levelId)
    RETURN 0

    SELECT @tempLevel = parentLevelId
    FROM Levels
    WHERE id = @tempLevel
END

INSERT INTO @filterIds
    SELECT filterId
    FROM Users
    WHERE id IN
        (SELECT substitutedId FROM Replacements
         WHERE SubstituteId = @userId
         AND (startDate <= @date OR startDate IS NULL)
         AND (endDate > @date OR endDate IS NULL));

INSERT INTO @allowedTable
SELECT id
FROM FilterItems
WHERE filterId IN (SELECT id FROM @filterIds)
AND filterType = @allowedType;

SET @tempLevel = @level

WHILE @tempLevel IS NOT NULL
BEGIN
    IF EXISTS (SELECT * FROM @allowedTable WHERE levelId = @levelId)
    RETURN 1

    SELECT @tempLevel = parentLevelId
    FROM Levels
    WHERE id = @tempLevel
END

RETURN 0

Listing 3.14: Check a filter

3.4.4 Setting up the demo environment for DW Access

To make sure that the evaluation with the end user was possible some implementation was also done in DWA. This was done by using the code for the user version of DWA and modify
3.4. Implementation

the model according to the one proposed in Figure 5.1. DWA is written in C# and all data is stored in a MSSQL database.

The main difference is that the model for DWA is using the data directly from the data warehouse as stated in section 4.1.1. This means that some functionality must be written to the specific data warehousing system. A demo environment for DWA was already up running this so the implementation meant to make relevant changes in this environment according to the model from the design phase.

The priority was to implement the things in the requirement specification (Table 5.2 on page 33) that were needed for the evaluation with the end user. This meant that things that had no direct impact on the user experience of DWA were not prioritized. The following was prioritized:

- Start and end time on filter items.
- Templates.
- Both user and group filtering in the same software.
- Replacements

Things that were done in this phase:

- Added the following tables:
  - Generic filter items.
  - Replacements.
  - Templates.
- Gave all users, groups, templates and roles a filter ID.
- Deleted old table for user filters that was based on user ID instead of filter ID
- Columns for start date and end date were added to the filter items.
- Changed the view for showing filters for users. The new view for filter is based on the filter ID and takes the date into consideration. The old one filtered on user ID instead.
- Changed procedures to use new tables and views instead of the old ones.
- Created stored procedures to load a template to a user, group or role.

3.4.5 Implementing into DW Access

This phase was to connect some of the database changes made in section 3.4.4 to DWA. Here the wireframes (Figure 5.2, 5.3 and 5.4) was the base for the user interface (UI) even if some small changes were done because of things that were not implemented for evaluation.

Generic filter items

Instead of storing a user ID or a group ID in the database table with filter items a generic filter type is used. It is the same approach as in the implementation for the demo database in section 3.4.2.
3.5 Evaluation

Merged versions for groups and users

This part was implemented by adding a tab with groups below the tab with users in the user version of the software. This is the same way as showed in Figure 5.2. This was implemented together with changes that made all filtering on the filter ID. First step was to adapt the code for user filtering to this approach. After this the generic filter items made the group implementation pretty much the same. Some differences were required though as a group item and a user item differed. The UI for this can be seen in Figure 5.8.

Templates

Templates were implemented by adding an option for adding a template in the properties of users and groups. The view for creating and editing a filter is the same as the one to change the filters for a user or a group. Templates are assigned a filter ID in the same table as a user or a group. This is the same approach as in the design model. The UI can be seen in Figure 5.8.

Replacements

The replacements were added with a UI very similar to the one suggested in the wireframe in figure 5.4. Except for the UI everything about replacements is handled by the database. This is working in the same manner as in the demo database. The UI for this functionality can be seen in figure 5.9.

Start and end dates

Start and end date was not fully implemented in the UI but the database was prepared with the necessary columns. Also the procedures was adapted to this feature. This was to evaluate that the procedures worked as expected. The parts that was implemented of this features was the one for replacements. When creating a new replacement a start date and an end date may be set in the UI.

3.5 Evaluation

3.5.1 Testing

Demo environment

The testing on the demo environment was performed with a black box approach where test data was added to the database and queried with known results were asked to the database. The result was compared with the expected result.

DW Access

For DWA the actual implementation of the demo environment was tested. This was done by the same black box approach as with the demo environment for the filter database. It was done by adding custom data into the database to see if it worked in the UI. Also adding data in the UI to see if the correct data was added to the database was tested.

3.5.2 User interface evaluation

The evaluation of the UI was performed by using the demo environment for DWA. The users had some time to perform some tasks in the UI and then answer a survey about their experience. Both more experienced users and not that experienced users were asked and the result was summarized. The tasks were the following:
3.5. Evaluation

- Set a filter to a user.
- Set a filter to a group.
- Add a template.
- Assign a template to a user.
- Create a replacement without start date or end date.
- Create a replacement with a start date and/or an end date.

The result of the survey can be found in Table 5.3.
In this section the software that will be discussed and analyzed in this thesis work is introduced. The text will focus on the access control features in the software and most of the other features will not be introduced. The first section will feature a short introduction about the different software. Section 4.3 will include a comparison table based on some use case scenarios that will be introduced in section 4.2.

4.1 Introduction

In this section the reference system DWA will be introduced. The other software that is part of the comparison in this chapter will also be introduced, but more briefly. As the documentation for DWA is very limited most of the information in the respective section about it will be knowledge from experimenting with the software and knowledge reported by developers of the software.

4.1.1 DW Access

DWA is a software developed by Pdb and it was the reference system in this thesis work. DWA is developed in two versions, one for administrating on an individual basis and one for administrating groups of user. In the rest of this section the word user will be used interchangeably between user and group.

It is sold as a service to companies and the actual version differ between different companies. It provides a central access control administration tool for data layers.

User Interface

The UI contains of multiple tabs with different functionality. One tab is for filtering. One user is selected in a tree view and filters can then be selected in another tree view. If a filter is set it means it is set for all underlying layers in the tree. If that is not what is wanted the administrator needs to set the filtering on a lower layer. This means that it is very flexible how fine-grained the filtering shall be. The other tabs is for administrating the databases and data areas for the solution. An example of the view for set filters for groups can be seen in Figure 4.1.
4.1. Introduction

Figure 4.1: The UI for setting filters for groups in DW Access.

**Architecture**

The software is built in a layered approach with a layer for the UI, one for the business logic and one data access layer for actually communicating with the data source. This architecture is shown in Figure 4.2.

**Implementation**

- DWA is implemented for Microsoft Windows in C# and a database built in MSSQL. The application writes filter settings to the database.
- The OLAP cube uses a dynamic-link library (DLL) file to read the filter settings from this database.
- The users are loaded into the database from AD using LDAP.
- The data access layer reads and writes to the database and returns the results to the UI via the business layer.
- The data is collected directly from the data warehousing system via views in MSSQL. This means that data is not required to be added explicit in DWA.
- Some tables and functions in the database are implemented specific for the underlying data warehousing platform.
4.1. Introduction

Figure 4.2: The different layers of DW Access and how they communicate.

- **DWA** uses four different layers for data representation:
  - Categories
  - Data areas
  - Dimensions
  - Levels

- Every data layer can be seen as detail level and data areas are sublevels to categories, dimensions to data areas and levels to dimensions or levels. The depth can be arbitrary because there can be an unlimited depth of levels. An illustration can be seen in figure 4.3.

The actual filtering is done in the data warehousing system. When a user is requesting data from the data warehouse it will request the filters for the user from **DWA**. This flow can be seen in Figure 4.4.

4.1.2 Jet Data Manager

Jet Data Manager is a software for transforming and validating data. It is a more complete solution for data warehousing than **DWA**. But it still has some good features for access control. This section will focus on the access control part of the software [28].
4.1. Introduction

Figure 4.3: The different data layers of DW Access.

Figure 4.4: The flow when a user is requesting filtered data in DW Access.

Usage
Jet Data Manager lets the user handle security in OLAP databases. The administrator can set access to different objects in the database. Access can either mean read or write. The access control is based on roles and users, that can be imported from AD. It can either use an approach where the administrator chooses what specific areas the users shall be able to see or a model where the administrator chooses what the users shall not be able to see.

Functions
Some functions in Jet Data Manager that are worth mentioning are:

- Inherited permissions on cubes.
- Read and write permissions.
- Synchronize groups via AD.
• Multiple environments.
• Create views for the data.

4.1.3 Timextender

Timextender Data Warehouse Automation (TX DWA) is a software for data warehousing.

Object level

TX DWA sets permissions based on the roles of the user in a MSSQL database. It gives the user the same access to the data as it should have in the database even in the context of data warehousing. There are three different types of access that can be set:

Allow Allow user if not explicitly denied by another role membership.
Deny Deny even if allowed by membership in another role.
Not set Deny if not allowed by membership in another role.

This is the same type of allow and deny that MSSQL uses [29].

Data level

It is possible in TX DWA to filter data for individual AD users and or MSSQL roles. By this feature the administrator can filter data on values of specific columns, RLS. For example let a user see sales data from some specific regions [29].

Views

It is also possible to do access control with the help of views. The views let the administrator create virtual tables with exactly the information he or she wants the user to be able to see. A virtual table can mean for example merging tables or filter data in a table without changing the data structure. The user then only gets access to the view and not the original table [29].

4.2 Functions

The functionality that will be considered in the comparison is described in this section.

4.2.1 Row-level Security

The functionality to set the filtering based on the value of some column/columns. This means that the filtering will be on row level and only show the rows that is within the rules of the filtering.

4.2.2 Views

The function to create views to decide how the data shall be shown to the user. It can for example mean filtering of data, combining content of tables, hide columns or many other ways of how the data is shown to the users.

4.2.3 Read/Write

This is the functionality of deciding if the user can change or only see the data.
4.2.4 Deny
A level of permission where the user is denied access. This means that even if allowed somewhere else it is still denied. In this case there also is a not set value where the user has no access if not allowed somewhere else.

4.2.5 Time limited filters
This is the function to set a start and/or end time for a specific filter setting.

4.2.6 Replacements
This is the function to set a user as a replacement for another user. The user will then have the same permission as the user it is replacing.

4.2.7 Switch between users/groups/roles
This means that the software can handle the filtering on different levels in the same software. For example let a user be a part of a group but still have some own settings.

4.3 Comparison
A table with a comparison of the different software presented in this chapter is presented in table 5.1. Also the proposed design for the new version of DWA presented in this thesis work is included in the comparison.
This chapter will present the results. It provides figures and tables. No discussion about the results is presented in this chapter. The discussion is instead found in chapter 6 where both the results and the method is discussed.

### 5.1 Pre-study

The result from the pre-study varied based on unique business requirements. But the things that were mentioned to be interesting were the following:

- Replacement functionality.
- Start and end time on filters.
- Access review functionality.
- Better processes for adding new users.
- Information classification.
- Possibility for users to request access.

### 5.2 Comparison

The result from the comparison part is presented in Table 5.1. The empty fields in the comparison was information that was not found for that specific software.

### 5.3 Design

Table 5.2 shows the requirement specification for the final design. It has three levels of priority where 1 is the highest and 3 is the lowest. Priority 1 means that it shall be included in the resulted design. Most of the priority 2 requirements shall also be included. Requirements with priority 3 shall only be fulfilled if there is enough time.

The model for the proposed filter database is presented in Figure 5.1. It is a design model for a filter database and is not adapted to DWA. The parts that have been actually implemented
Table 5.1: A comparison between the different software presented in the report.

<table>
<thead>
<tr>
<th>Software</th>
<th>DWA</th>
<th>JDM</th>
<th>TX DWA</th>
<th>New DWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row-Level Security</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>View</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Limited</td>
</tr>
<tr>
<td>Read/Write</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Deny</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Time limited filters</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Replacements</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Switch between users/groups/roles</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Requirement specification

<table>
<thead>
<tr>
<th>Description</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filters on both user and group basis</td>
<td>1</td>
</tr>
<tr>
<td>Add support for MSSQL 2016</td>
<td>1</td>
</tr>
<tr>
<td>Add support for Azure SQL</td>
<td>1</td>
</tr>
<tr>
<td>Support for older versions of MSSQL</td>
<td>2</td>
</tr>
<tr>
<td>Time limited filters</td>
<td>2</td>
</tr>
<tr>
<td>Replacements</td>
<td>1</td>
</tr>
<tr>
<td>Templates</td>
<td>2</td>
</tr>
<tr>
<td>Read and write permissions</td>
<td>3</td>
</tr>
<tr>
<td>See all permissions for individual users</td>
<td>1</td>
</tr>
<tr>
<td>Deny feature</td>
<td>1</td>
</tr>
<tr>
<td>DLL-modules for data access layer</td>
<td>3</td>
</tr>
</tbody>
</table>

in the database for DWA can be found in section 5.4. A list with short descriptions of the tables is presented here:

Level A table with the objects in the system.

FilterType The different rules of the filters.

FilterItem The filter entities.

Filter A gateway table to the subject of the filter.

Template A table to store templates that can be added to other subjects.

User A table with all users in the system.

Group A group represents an arbitrary number of users.

Membership The table that connects users to a groups.

Replacement The table to store if a user is replacing another user.

Relation A table to store relationships between the users for ReBAC.

Relationship A table with different types of relationships.

The wireframes for the design of the proposed changes in DWA is presented in Figure 5.2, Figure 5.3 and Figure 5.4. They represent the UI for the new features for setting different filters, time limits, combined user/group filtering and the replacement feature.
5.3. Design

Figure 5.1: Entity-relationship diagram for the proposed model.

Figure 5.2: The new view for filtering.
Figure 5.3: The right click menu on a filter item.

Figure 5.4: The view for setting up or edit a replacement between users.

5.4 Implementation

This section includes flowcharts for some of the functionality implemented in the demo database. It also includes some screenshots on DWA with some of the implementations that has been done.

- In Figure 5.5 the flowchart for adding a user or a group can be seen.
- In Figure 5.6 the flowchart for adding a template to a user or a group can be seen.
- In Figure 5.7 the flowchart for checking if a user is allowed to see a requested level can be seen.
- In Figure 5.8 the UI for the merged version with users, groups, templates and roles can be seen.
- In Figure 5.9 the UI for adding a replacement can be seen.
5.4. Implementation

Figure 5.5: Flowchart for adding a user or a group.

Figure 5.6: Flowchart for adding a template to a user or a group.
5.5 Evaluation

The result from the survey in the evaluation can be seen in Table 5.3. The answers are graded from 1 (strongly disagree with) to 5 (strongly agree with). Only experienced users were asked to answer the question if it was an improvement or not.

Table 5.3: Result of the survey in the evaluation part.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>It was easy to set a filter for a user</td>
<td>0 0 0 1 3</td>
</tr>
<tr>
<td>It was easy to set a filter for a group</td>
<td>0 0 0 1 3</td>
</tr>
<tr>
<td>It was easy to create a template</td>
<td>0 0 0 1 3</td>
</tr>
<tr>
<td>It was easy to assign a template to a user</td>
<td>0 0 0 2 2</td>
</tr>
<tr>
<td>It was easy to create a replacement without time limits</td>
<td>0 0 0 0 4</td>
</tr>
<tr>
<td>It was easy to create a replacement with time limits</td>
<td>0 0 0 0 4</td>
</tr>
<tr>
<td>I would consider to use this version of DWA</td>
<td>0 0 0 1 3</td>
</tr>
<tr>
<td>I think this is an improvement for the functionality of DWA</td>
<td>0 0 0 0 2</td>
</tr>
</tbody>
</table>
5.5. Evaluation

Figure 5.8: The UI for filtering in the merged version. Also the templates can be seen.

Figure 5.9: The UI for creating a new replacement.
6 Discussion

This discussion chapter will be divided in two parts. In section 6.1 the results from the thesis work are discussed. In section 6.2 the method is discussed.

6.1 Results

6.1.1 Pre-study

The pre-study showed that the need for a tool for administrating access control like DWA was requested. It also showed that DWA was still used. It confirmed that new functions were requested by the existing users. Users had also looked at the possibility to use other systems instead of DWA. That confirmed that a comparison with other software for access control was of high interest, too. The result from the pre-study was mostly as expected. Although some things not expected were discovered. One surprising fact was that there seems to be such a lack of similar software to DWA. Also some very common tasks at larger companies had no really easy solution in the software. An example of this are the lack of replacements, both temporary and permanent, in all software that was found. But the results from the pre-study was mostly expected.

Overall, the pre-study showed that it was worth going on with the thesis work as planned. It showed that it is in an area where there is much unfulfilled potential.

6.1.2 Comparison

The first fact to be mentioned is that DWA is a product that has not that many real competitors. The functionality is often a part of larger data management systems. That was one of the first thing that was discovered during this thesis work. DWA is a tool specified to handle access control independent of the data layer. The rest of the discussion in this section will be about table 5.1. The table shows us that DWA lacks a lot of functionality compared to its competitors. Things like the distinction of read and write permissions, a different model of permissions compared to MSSQL showed that the lack of functionality was significant. It can also be seen that most of the competitors have similar functionality and that some of the functions that are asked for to be added in DWA are absent in the competitors. So this step gave the information that it can be worth investing more into DWA even if the product
today lacks some significant functions. It is also the only one of the compared software that provides standalone access control.

6.1.3 Design

The design phase was mostly about doing the right limitations according to the limited time. Many ideas did not make it to the design due to this. The motivations for the decisions can be read in the following list:

- Functionality to replace a user had been asked for before the start of this thesis work. The functionality had neither been implemented and except from the status report no evaluation of it was done by Pdb. None of the other software compared in the thesis work had this functionality. So this was considered a highly interesting aspect to consider in the design.

- All software that was considered as competitors made a distinction between permissions for read and write. It seems like this really is worth the relatively small effort to implement. This to make sure that DWA is considered as a good alternative for any user that is searching for an access control system.

- A logging feature was not included. Although Ravi S. Sandhu et al. consider that it is a very important part of a complete access control tool it was considered outside the scope of this thesis work.

- Limit the amount of accesses to objects. It was not included in any competing software. After consideration it seemed like something that was of low interest for the target group. It could be interesting for some sort of paid service but that is not the target group for DWA.

- To merge the models and let access be controlled on both user level and group level was a core of the thesis work. It was also the highest priority from Pdb. As this meant a lot of interesting design work it was considered as highly interesting in this thesis work.

- The deny feature is included in MSSQL. One of the limitations to the thesis work was to focus on that platform. Therefore it seemed like a good idea to use a similar approach in DWA. The feature was also found at all competitors.

- Just one type of replacement policy is available in the model. It would be a reasonable work to extend this by allowing different types of replacements. This would increase the complexity of the thesis work and it was therefore considered but not a part of the design.

- In the design suggestion only a limited version of views is suggested. The competitors included the feature to make views in the database. It seemed logical because the competitors handled the data storage also. The access control is not the main feature as it is in DWA. As the design is for standalone access control though only limited view functionality is added. The solution is to provide RLS in data layers that do not support RLS.

- In the database design support for relationships is included. Even if it is not a highly prioritized part of the design and no implementation thoughts about it were considered it shows that the model is extendable.

- No records are ever deleted from the database. To delete a filter item it sets the end date to the current time. When a new filter item is created it sets start date to the actual time if not explicitly set. Only filter items from older versions of the filter database will have NULL as start date. The reason to save all entries in the database is that it is possible to see what filters a user had at a specific time if necessary.
6.1. Results

Another approach could be to save all old records in a database with historic data. The right approach is dependent on scaling issues and complexity. This approach would probably be faster with a lot of data. But if the amount of users or data areas are not very large that should probably not be a problem. But there was not enough time to do testing about how well it scales. The advantage for this approach is that it simplifies the implementation. In this thesis work it was considered of higher value than slightly faster computations.

- Information classification was not considered. Information classification would require deeper integration with the object handling. Therefore it was not processed in this thesis work because the time was not enough. It would be a very good thing to look at it in the future though.

- In MSSQL 2016 RLS could be used. This gave the expected result. This means that it would be possible to connect the filter database to a database in MSSQL 2016. Another approach is to have the filter table in the same database. The function for the RLS would then be used to read from the filter database. This was not implemented but should be possible.

6.1.4 Implementation

The implementation part showed that it was possible to make a working prototype by the model developed in this thesis work. It also showed that is possible to convert this prototype into a real system even if this step requires more complexity. The increased complexity is because adjustments to the specific system must be considered. In DWA this meant that adaptations because of the integration with the data warehousing system had to be done.

One of the strengths with DWA was that it has well structured layers. The layered approach is mentioned in the architecture part of section 4.1.1. This meant that the workflow to implement the adaptations to the software was pretty structured. The result is still a prototype though. Some of the changes required to put it on the market are the following:

- Tighter connection with AD. Even if some adaptations were done the time was not enough. Therefore the focus was to make the implementations in the UI and the database. That was enough to evaluate the system for this thesis work.

- Test with larger amount of data to make sure it scales well.

- Try to connect the filter database to a security policy for RLS in MSSQL if it shall work on more data sources.

6.1.5 Evaluation

Testing

The testing showed that the implementation both in the demo database and DWA provided the requested functionality. This means that it should be possible to build an access control system around the provided model. Of course adaptations specific to the end users requirement are required.

User evaluation

The user evaluation shows that the implemented functions does not increase the complexity that much. All testers agreed or strongly agreed that the new implemented functions were easy to use. All users would also consider using the new version of DWA if they considered a tool for access control. All experienced users thought it seemed like an improvement compared to old versions.
The summary of this result is that the functionality that has been implemented as a part of this thesis work makes DWA a better contestant on the market. This both according to experienced and not experienced users.

6.2 Method

6.2.1 Pre-study

As DWA is not a widely used software the number of possible users that could give input was not as many as preferred. But the ones that were asked gave very insightful information. Material that Pdb had from an earlier evaluation gave even more insight in the situation. That meant that the drawback of the lack of users was not of that dignity that was first thought. This stage of the work gave good insight what the end users were expecting.

6.2.2 Comparison

Here the biggest drawback was that most of the competitors to DWA were costly and they could therefore not be tested in this thesis work. Although this could be a big drawback the documentation for the software was enough to understand the functionality in the software. But of course it would be preferred to actually test them and get a more end user view of the functionality. Also some open source alternative in the comparison would be preferred. This would make it possible to examine the code, of course a huge advantage. But DWA should probably be compared with proprietary software as it neither is open source. So using documentation for proprietary software seems good enough. Also a narrower definition of a competitor could be used. It would then be hard to actually find any competitors as mentioned in subsection 6.1.2. The pre-study gave good insight about which features that were interesting to compare.

6.2.3 Design

After the earlier steps the focus for the design was set. So well known tools as ER-diagrams and requirement specifications were used for this step. Inspiration was also taken from actual designs and design proposals provided by Pdb. Also a development database was available. That meant that ideas could be tested at a pretty early stage. The thought behind the design is that it should fulfill the requirements for this thesis work. Another huge focus of the design phase was to secure that it was extendable after the end of this thesis work.

6.2.4 Implementation

The demo database was a simplified version of the actual database. That means that a implementation in a real system will be more complex than this simplified version. This simplification made it possible to try more ideas during the limited time of the thesis work. Some of the functionality was also converted to a real system based on DWA. This was mainly to make sure the design was working in a real world situation. This lead to some interesting adaptations because of the slightly more complex system for levels and data areas used by DWA. The reason for this is that DWA is connected directly to the data warehousing database. The demo data base is instead a completely stand alone system. The basis of the two approaches was very similar though.

The solution with RLS in MSSQL 2016 was never implemented in DWA. It would be possible by making security policies connected to the filter database. Also some view for loading the data areas and users from the database to the filter database would be required. This would not give any changes in the UI and would not mean very much in the evaluation. So the implementation for the filter database was considered enough for the evaluation of
6.3. Thesis work in a broader context

the software. It was considered as enough to test RLS in MSSQL to evaluate the functionality. That was decided to be enough testing of that feature.

The solution for older versions of MSSQL may be unnecessary. Here a better solution may be to migrate the system to a newer one. A custom system may be a better solution in the specific cases where older versions are required for some reason. This question requires a deeper comparison of different versions of MSSQL and is not answered in this thesis work. It is instead left as a subject in the future work chapter.

The parts with relationships was never implemented even if it was a part of the provided model. The reason is that even if relationships have been considered and are part of the design it has not been unambiguously defined. As stated multiple times earlier the access control needs can be very unique and the relationship model needs to be implemented accordingly to the needs of the specific end user. Some models that could be considered are a manager role that can see all data any of their employees can or a friend relation where a user can see every object that the other user is the owner of. The latter will of course require that the system knows who the data owner is which is not the case in the provided model.

6.2.5 Evaluation

Testing

As the main purpose of the implementation was to evaluate it by the end user, a black box approach for the testing seemed good enough. Of course unit testing would be good. But it was not a prioritized task due to the limited time of this thesis work. This was because the main purpose was not to provide perfect functionally as mentioned earlier. Of course some testing of the code was done as it was written but not in a structured way enough to call it a white box approach.

User evaluation

The user evaluation was done both with experienced users and not that experienced. The reason for this was that input from new users gives another perspective on the software. This perspective is good because experienced users are very familiar with the software and have a clear opinion already. Although the experienced users’ input is of course important as they will probably use the software in the future. That is why a mixture was decided to be the best reference group for this survey. Also the reference group would have been very small if only experienced users were asked. This is one of the downsides when working with such a niche product. The user base of the software is small. Although the answers were very unambiguous so the conclusions from the survey was clear.

6.3 Thesis work in a broader context

In this section the work will briefly be analyzed by consequences on the society and ethical aspects. An important aspect of this work is that some sort of access control may be required by legislation, for example in the healthcare industry. This means that access control is required by the society. But even when it is not required by legislation it may still infringe on the personal integrity. But of course there are downsides also. What will happen if a doctor is not allowed to read the record of a patient for example if the ordinary doctor is by some reason absent. This is one case where the replacement feature can be useful.

One difficult question regarding the ethical aspects of access control is when it is not obvious who the data owner is. One example of this is if an organization is collecting sensitive personal information, is it the organization or the involved people that shall be considered data owners? Can one individual prohibit an employee at the organization to access the personal data? This are questions with no obvious answers. This are also questions that this thesis work has not tried to answer but it is important to understand the consequences when
6.3. Thesis work in a broader context

using a system. Access control helps protect the sensitive information without blocking required access, as long as it is implemented with a well-defined policy and sufficient control mechanism to make sure the policy is followed. The system itself shall be considered as a tool that will only cause problem if it is used in the wrong way.
7 Conclusion

In this section the number for the conclusion is representing the research questions introduced in section 1.3.

1. The thesis work shows how to build an access control system for hybrid solutions. Most data layers have huge similarities. Old versions of MSSQL that were briefly investigated in this thesis work were the most clear exception. But even there it should be possible by creating views connected to a filter database similar to the one used by DWA today. The biggest problems is instead the different needs from different end users. But if they have clear requirements it should be possible to design a system for most of their needs. The data layer does not seems to bring any huge problems.

2. Today the situation is that access control is most often implemented as parts of other systems. Stand-alone software in the area is very scarce. The systems are pretty basic and are mostly implemented around pretty straight forward RLS. This means that the market is open for more advanced niche products.

3. A good system for access control shall not be noticed by the users of the system. Only the system owner and assigned administrators shall be required to know it. By using the approach that for example DWA uses the users will not notice it. The way it is implemented means that they can handle the data in the same way as if they had full access. The only difference is that they get filtered data. One exception of this may be when a user is trying to access data that is not permitted. But if that is noticed it is not by the access control layer. Instead it is because of the UI implemented in the actual data handling software.

4. The future is very much about adjustments to the systems already existing. But access control is also an area in constant change. One example is that the increased numbers of users on social media platforms have opened up new areas to look at. One example is ReBAC where the user relations is used to make access control decisions [8]. Several interesting questions have raised from this. For example what will happen when a user is allowing a friend’s friend to see potentially sensitive data? Do the user really know what will happen when delegating this decision to their friends? An interesting extension on this is how this thoughts can be implemented into business relationships. The model from this thesis work has implemented the concept with relationships. But
how it will be used is a completely different question that can lead to further important conclusions in the area.
This thesis work has some limitations introduced in chapter 1, and motivated in chapter 6. Also there is possible work that is outside the scope of this thesis work. The potential for future work in the area is huge. Section 8.1 describes work in DWA that can be done and section 8.2 considers more generic questions to an arbitrarily system. Some of the questions were asked already in chapter 6.

8.1 DW Access

One of the purposes of this thesis work was how to design an access control system that worked independently of the data layer. Some ideas are in the provided design. They are not implemented in the evaluation version of DWA though. It would be interesting to adapt the data access layer in DWA to the proposed solutions and try them in a real system.

Another thing that would be interesting is classification of objects that was discussed in chapter 6. How would the design and the complexity of the system be affected?

Also it would be interesting to build a user interface for the users of the system. Features in this interface could be requests for access to some objects. Another example could be the possibility to see own filters. Maybe setting permissions for own objects could be a possibility in this user interface?

8.2 Generic

The provided design is not tested enough in a real world situation. How well does it scale? Would it be better to use a historic database in a real world situation? Testing in the environments at the end users would give many interesting answers. So this seems like the next phase in this thesis work.

Another interesting thing would be to evaluate how the relationships would look based on business relations. This would maybe lead to a more ambiguous definition of the term relationship than the one provided in this report. It would also include a consequence analysis. How would the relationships affect the confidentiality, the integrity and the availability of the data?
The last question in this chapter is if it is really necessary to build solutions for old software. It may be better to migrate users to later versions of the software instead. This is a very relevant question because of the features for RLS that was introduced in MSSQL 2016.
Bibliography


